The Deployment of a Constraint-Based Dental School Timetabling System

Hadrien Cambazard, Barry O'Sullivan, Helmut Simonis

Proceedings of the Twenty-Fifth Innovative Applications of Artificial Intelligence Conference

The Deployment of a Constraint-Based Dental School Timetabling System

Hadrien Cambazard
G-SCOP
Université de Grenoble, France
hadrien.cambazard@grenoble-inp.fr

Barry O'Sullivan
Cork Constraint Computation Centre
University College Cork, Ireland
b.osullivan@4c.ucc.ie

Helmut Simonis
Cork Constraint Computation Centre
University College Cork, Ireland
h.simonis@4c.ucc.ie

Abstract

We describe a constraint-based timetabling system that was developed for the dental school based at Cork University Hospital in Ireland. This system has been deployed since 2010. Dental school timetabling differs from other university course scheduling in that certain clinic sessions can be used by multiple courses at the same time, provided a limit on room capacity is satisfied. Starting from a constraint programming solution using a web interface, we have moved to a mixed integer programming-based solver to deal with multiple objective functions, along with a dedicated Java application, which provides a rich user interface. Solutions for the years 2010, 2011 and 2012 have been used in the dental school, replacing a manual timetabling process, which could no longer cope with increasing student numbers and resulting resource bottlenecks. The use of the automated system allowed the dental school to increase student numbers to the maximum possible given the available resources. It also provides the school with a valuable "what-if" analysis tool.

Introduction

Universities and hospitals are under considerable pressure to reduce costs while improving service delivery. A central component to this effort is the availability of timetabling systems that can find practical solutions to maximizing the utilization of teaching resources, such as facilities and staff, while not compromising on education quality.

Traditional university timetabling is often concerned with the task of assigning a number of events, such as lectures, exams, meetings, and so on, to a limited set of timeslots (and perhaps rooms), in accordance with a set of constraints (Cambazard et al. 2004). Three main classes of the university timetabling problem have been identified: school (Kingston 2012), course (Cambazard et al. 2012) and examination timetabling (Burke et al. 2012). A fundamental constraint appearing in all the problems is the "event-clash" constraint. This states that if a student is required to be present for a pair of events (for example courses), then these must not be assigned to the same timeslot, as such an assignment will result in this student having to be in two places at the same time. This particular constraint can be found in almost all university timetabling problems (Bonutti et al. 2012; McCollum et al. 2010). The problem restricted to these constraints alone can be viewed as a *graph coloring problem*. This is a pure and difficult combinatorial problem in its own right and can often become more challenging when combined with the many side constraints occurring in the context of timetabling. Most university timetabling involve a hard graph coloring sub-problem and current timetabling systems are tailored to address this recurrent and common pattern. However, this pattern is not the main source of difficulty in many medical and dental school timetabling settings.

While the dental school problem also involves a coloring sub-problem, this sub-problem is unlikely to be the most challenging feature of the problem because the number of students and events is much smaller than in traditional university timetabling problems. The difficulty in dental school timetabling arises as a consequence of resource availability, such as specialist equipment and seating, which are very expensive. Most universities cannot afford to relax these constraints by increasing resource capacity, so they rely heavily on good quality timetables. This resource utilization defines a *bin packing problem*. The packing comes from the fact that, in medical/dental school timetabling, we are trying to find a timetable where the resources, e.g. dental chairs, are not left idle. Commercially available timetabling systems do not deal with dental school timetabling very well primarily because such systems are designed to exploit hard coloring subproblems rather than bin packing. Despite its unique challenges, the timetabling community has not focused effort in this direction, so the work presented in this paper is novel from both scientific and applications perspectives.

The Deployed Application. In this paper we present a constraint-based dental school timetabling system that has been in use at the dental school at the Cork University Hospital since 2010. The system was developed at the Cork Constraint Computation Centre (4C) in close collaboration with the management and staff of the dental school. In 2010 the dental school was facing a significant increase in its student intake from approximately 40 to 50 students across all years of its programe. While the timetable up to that point was created by hand, the dental school could not find a satisfactory timetable due to the increased resource restrictions imposed by the increase in student numbers. In the years 2010 to 2012 the first class of increased student numbers

had a growing impact on the school's timetable; this process will conclude in 2013 when all classes will run with the increased student numbers. The impact of this was that during the transition period the timetable of each previous year was of limited use, precluding the normal process of updating the existing schedule slightly from year to year. A new approach was needed. This paper presents the timetabling system that was developed and discusses the experiences of its deployment over the past three years.

The remainder of this paper is organized as follows. We first informally describe the problem and introduce the required notation. We then describe the solution process and alternative choices for solvers and interfaces. A constraint-based model of the problem will then be presented, along with an example of its output. Finally we will discuss the evolution and maintenance of the system over the three year deployment period to-date.

Problem Statement

We describe the problem solved by the application in an informal way, while introducing some notation for the formal presentation which comes later in the paper.

Time Slots

The timetable is generated for each term, and all weeks in the term run with the same schedule. We, therefore, deal with generic days (set D) from Monday to Friday, with three time periods (AM, midday, PM) scheduled for each day. We, thus, assign 15 time slots (set T) in a week. For each time slot, we have a period weight p^t_{cost}, which indicates possible preferences to using the time period. At the moment, all midday time slots have a cost of 1000 (strong preference against using these periods), and Monday morning and Friday afternoon have a cost of 1 (weak preference against using them). All other time slots have a cost of 0. The function onDay_t indicates the day to which time slot t belongs.

Student Groups

The students of each year are organised into groups comprising 7-10 persons (set G), which are allocated as a unit in the timetable. Naturally, each group can only follow one course at a time. When we started in 2010, there were four groups in each year, this has been increased progressively to five to increase student numbers. Note that the groups do not all have the same size, and the group sizes change from year to year, as students repeating a year disappear from one group, and are assigned to another group in another year. Table 1 shows the group sizes for the year 2012, note that Year 5 is still using the lower student numbers of the old system, while Years 3 and 4 already use the increased numbers. Also note that it is difficult to increase group size beyond 10, as several labs have a capacity limit of 10.

It is possible to have multiple groupings for the students, e.g. to have smaller groups for clinics, and larger groups for lectures or seminars. Two such groups that share students cannot be allocated to the same time slot; this is expressed by *Incompatibility Time Slot* constraints, described below. At the moment, this feature, while implemented, is not used in the actual timetable.

Table 1: Classes and Group Sizes

Class	Group Sizes
Hyg	16
Y3	10 10 10 10 9
Y4	9 10 10 10 10
Y5	8 8 8 9 9

Subjects

The subjects that are allocated in the timetabling system are all related to labs and clinics. Unlike most other courses at universities, the location for these labs and clinics is automatically given, as only one room for each has the necessary equipment. Therefore, we are not concerned about room allocation. Also unlike labs in other disciplines, it is possible to have groups from different years in the same clinic at the same time, provided the room capacity is not exceeded. The room capacity may be defined by physical constraints (e.g. treatment chairs), or by the availability of teaching assistants, when maintaining a required student/teacher ratio. Table 2 shows the list of subjects (set S) and the capacity of the labs (function capacity_s) as the first two columns. In 2012, two seats were added to the Restorative Clinic room, bringing the capacity from 34 to 36. This clinic also contains four additional chairs (in general described by the function extra_s) normally reserved for private practice and post-doctoral work, which can be used if required. As they are located outside the main clinic area, this creates problems with supervision, and should only be done if this extra capacity is really needed. For every use of some extra capacity, a cost e_{cost} must be paid.

Especially in the first three years, students also follow theoretical courses in lectures and seminars. These courses are easy to assign manually and use other rooms, and are therefore not handled by the system.

Curriculum

The curriculum defines how many sessions per week in each subject must be scheduled for groups of each year. In our model, we use the function demand_{gs} to specify how many sessions group g must be scheduled in subject s. The current curriculum is shown in Table 2. Note that only the lab- and clinic-based courses are shown.

Table 2: Curriculum

Subject	Cap	Hyg	Y3	Y4	Y5
OTL	24	*1*	*2*	*1*	-
Restorative Clinic	36 (+4)	-	*1*	*3*	*4*
Pros Lab	20	-	2	-	-
Dental Surgery	10	-	-	1	1
Ortho	10	-	-	1	1
Paedo	10	-	-	*1*	*1*
C and B Lab	10	-	-	1	-
Study	18	-	-	1	1
Restorative Tutorial	10	-	-	-	1

Doubling Up

Usually, in a clinic each student works on his own treatment chair, supervized by teaching assistants and lecturers. It is possible to assign two students to the same chair, reducing the demand on chairs in the clinic, but at the same time decreasing the quality of teaching. For the third-year groups, this "doubling up" may be acceptable; in general this is indicated by a predicate doubleUp_g. Whether doubling-up is preferable to using extra chairs in a clinic is a user choice. The function size_{gu} gives the size of group g depending on whether they double up ($u = 1$) or not ($u = 0$). The set U denotes the two choices. For each use of doubling up, a cost d_{cost} must be paid.

Instrument Cleaning

For some of the subjects, the students must use their personal instrument sets. This is indicated by the Boolean function $\text{needsCleaning}_{gs}$. After use, these sets must be sterilized, which takes about 2 hours. This is easily achieved overnight, but if the instruments are required twice on the same day, the cleaning time reduces the time available for practical work. This should be avoided, if possible. Italic values in Table 2 show which courses require instrument cleaning. If a group has to wait for the cleaned instruments on some day, a cost c_{cost} must be paid.

Allocation Constraint

When the timetable was generated manually, it was easy for lecturers to request specific time slots for their courses due to other commitments or personal preferences. In order to allow these wishes to be added in our system, we allow optional constraints that force, forbid or restrict the assignment of specific courses to specific slots. These wishes are expressed in the user interface as tuples of groups, subjects, time slots and a constraint type, one of FORCE, FORBID, RESTRICT or DONTCARE. By carefully choosing the combinations of groups, subjects and time slots, we can minimize the number of rules that need to be given by a user. The DONTCARE value is useful to temporarily disable some allocation constraint during the exploration of a scenario, without having to re-enter the constraint later on again.

The allocation constraints are also required to handle the needs of the Hygiene student group, which uses the OTL lab, but whose timetable is controlled by another university department. After discussion with their timetable manager, a fixed time slot on a Monday morning is allocated for the OTL lab of the Hygiene group.

Incompatibility Subject

It is possible to restrict which groups work together in the same time slot for a given subject. This might be required if one teaching assistant is handling, say, both a Year 3 and a Year 5 group on a given topic. Then these groups should not be scheduled together at the same time, in order to maintain a good student to teacher ratio.

Incompatibility Time Slot

Our model allows for different groupings of students for different topics, e.g. larger groups for seminars and smaller groups for specific lab work. Thus, we must require that two groups sharing some students cannot be scheduled at the same time. As our model does not deal with individual students, we must specify rules for all groups of that form.

Occurrence

We can limit how many courses a group can be assigned to on any given day, e.g. state that a group should only have one lab on a Wednesday, as some other lectures have to be scheduled on that day. Note that there is some overlap with allocation constraints.

Solution Process

In this section we describe the changes in the solution approach we have been following over a period of three years.

Initial Approach

When we started the project, we did not know which solution technology would be best to solve the problem, and whether we would be able to solve the problem at all. We therefore started with an exploration of possible solver designs. A first version was implemented with the finite domain constraint programming system Choco.[1] In the model, the variables are the courses to be assigned, and the values are the possible time slots. Capacity constraints for the rooms are expressed by bin packing constraints (Cambazard and O'Sullivan 2010), while many other constraints turn into disequality or alldifferent (Régin 1994) constraints. While initial solutions for feasible problems could be found quite quickly, it was hard to prove optimality or to show infeasibility of over-constrained systems. A specific problem were the symmetry constraints due to groups in the same year, i.e. with the same curriculum, and identical group sizes. These symmetries cause a large number of equivalent solutions, but are not easily removed completely. Other symmetries due to repeated courses for the same group can be handled with inequality constraints. As an alternative, a MIP model for the basic problem was considered, using the CPLEX solver.[2] This model did not consider doubling up on courses, and did not have the cleaning constraints.

Operationally, the 2010 problem still had many specific preferences to force classes at specific time slots in order to simplify teaching resource assignment. Enforcing all of these preferences typically lead to an over-constrained problem. It was clear that the system should allow users to play with enabling/disabling of complete classes of constraints and/or individual constraints, in order to find a good compromise satisfying the different stakeholders in the process. In order to allow this experimentation, a Web-based user interface for the system was developed. It allowed different persons to evaluate timetables, and to play with specific constraints. The user interface model was spreadsheet-like, implemented in Javascript, while calling back-end solvers written in Java.

[1] http://www.emn.fr/z-info/choco-solver/

[2] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

Re-write as Application

For the timetable of 2011, we decided to rewrite the system as a dedicated Java application, with a complete graphical user interface. As new constraints were added, it became increasingly difficult to integrate those changes in the ad-hoc Javascript solution. A more flexible solution was required. For this we used an application framework under development at 4C, which supports the creation of a complete application from a basic data model and a user interface definition. This drastically reduced the implementation effort for this specific application, as most components were already provided by the framework. Instead of adapting the existing solver code, we re-implemented the solver using the new data model. Only the MIP-based model was maintained, as it was able to determine infeasibility of problems much more quickly than the Choco model.

We packaged the resulting application for the Dental School as a stand-alone Java program running on a laptop. In the Dental School an intern from the Computer Science department was responsible for creating and modifying potential timetables, getting feedback from the different stakeholders in the faculty. This process took several iterations, finally creating the timetable implemented for 2011.

Dealing with Under-Capacity

Figure 1 shows a screen-shot of the application, displaying the solution for year 2012. In 2012, we changed the process again. As both the original project sponsor and also the intern using the system for the timetable of 2011 were no longer at the school, we faced the task of either training yet another person in the use of the system, or to provide the timetabling as a service. As initial discussions showed that the constraint model needed significant changes to deal with the increased resource requirements, we found it easier to make these changes in the system, while at the same time providing intermediate solutions for the Dental School as text documents, generated by our application framework.

The main change required was dealing with systematically over-constrained problems. The total demand for the restorative clinic exceeded the available capacity, even with the increase of capacity from 34 to 36 chairs. We either had to use doubling up as a solution to reduce demand, or to use extra chairs to increase capacity. It was not clear which of those approaches would be more readily accepted by the faculty; we therefore provided solution examples for both methods.

Once it was clear that solutions for the overall problem could be created with these extensions, several additional changes of the timetable were requested by different stakeholders. For example, the solution should have Year 5 students in the restorative clinic in the morning, and Year 4 students in the afternoon, as this simplified the assignment of teaching assistants. This could easily be handled by adding some new rules for the allocation constraints.

Dealing with the manager of the timetable in the dental school directly reduced the number of interactions with different groups of stakeholders, while maintaining the existing process in the dental school. Overall, the time needed to build the final time table was minimized, while involving 4C as a partner, and improving the tool through the interaction and the new constraints required.

Current Constraint-based Model

We describe the current constraint-based model for the dental school timetabling problem, describing the variables, the objective function and the constraints used. All necessary notation were introduced earlier.

Variables

We describe the model currently used for year 2012. The key decision variables are binary variables x_{gstu}, which indicate if group g is assigned to subject s in time slot t, either doubled up ($u = 1$) or not ($u = 0$). We require $16 * 9 * 15 * 2 = 4320$ of these variables for our problem, a medium-sized problem, so that using a four dimensional array does not pose a problem.

$$\forall_{g \in G} \forall_{s \in S} \forall_{t \in T} \forall_{u \in U} : \quad x_{gstu} \in \{0, 1\}$$

Next, we introduce binary variables y_t, which indicate whether any course is taught in time period t. These contribute to the cost function.

$$\forall_{t \in T} : \quad y_t \in \{0, 1\}$$

We also use binary variables z_{dg} to state if group g will require instrument cleaning during day d. These variables also contribute to the cost function.

$$\forall_{d \in D} \forall_{g \in G} : \quad z_{dg} \in \{0, 1\}$$

Finally, we use a set of integer variables v_{st} which state if in time period t we use some extra capacity for subject s. The upper bound of the domain is given by the function extra_s, which will be zero for most subjects.

$$\forall_{s \in S} \forall_{t \in T} : \quad v_{st} \in \{0, \text{extra}_s\}$$

Objective Function

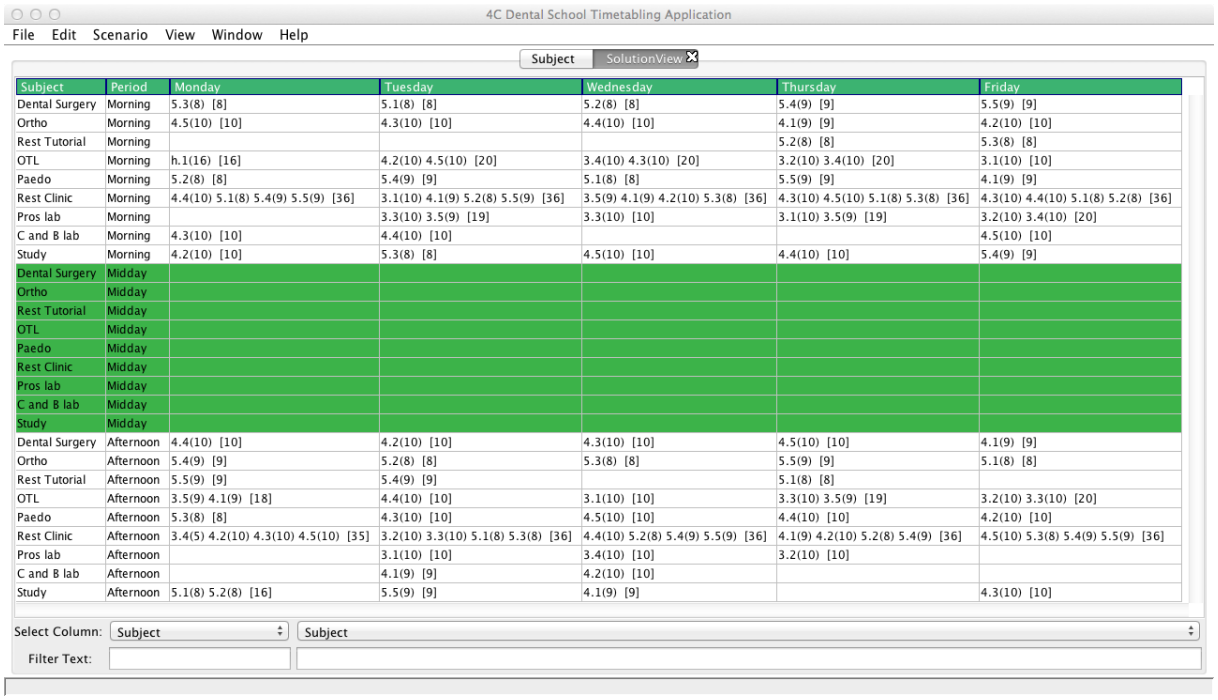
The objective function minimizes four elements: the total cleaning effort during the day in (1), the number of doubled up sessions in (2), the use of non-preferred time periods in (3), and the use of extra capacity (4). Note that we could easily introduce personalized costs for groups or subjects, if for example the loss of teaching time due to cleaning is more acceptable for year 4 than year 5 students. On the other hand, this would require more input data from the user, a change that should not be undertaken lightly.

$$\min \sum_{d \in D} \sum_{g \in G} z_{dg} c_{\text{cost}} \tag{1}$$

$$+ \sum_{g \in G} \sum_{s \in S} \sum_{t \in T} x_{gst1} d_{\text{cost}} \tag{2}$$

$$+ \sum_{t \in T} y_t p^t_{\text{cost}} \tag{3}$$

$$+ \sum_{s \in S} \sum_{t \in T} v_{st} e_{\text{cost}} \tag{4}$$



Subject	Period	Monday	Tuesday	Wednesday	Thursday	Friday
Dental Surgery	Morning	5.3(8) [8]	5.1(8) [8]	5.2(8) [8]	5.4(9) [9]	5.5(9) [9]
Ortho	Morning	4.5(10) [10]	4.3(10) [10]	4.4(10) [10]	4.1(9) [9]	4.2(10) [10]
Rest Tutorial	Morning				5.2(8) [8]	5.3(8) [8]
OTL	Morning	h.1(16) [16]	4.2(10) 4.5(10) [20]	3.4(10) 4.3(10) [20]	3.2(10) 3.4(10) [20]	3.1(10) [10]
Paedo	Morning	5.2(8) [8]	5.4(9) [9]	5.1(8) [8]	5.5(9) [9]	4.1(9) [9]
Rest Clinic	Morning	4.4(10) 5.1(8) 5.4(9) 5.5(9) [36]	3.1(10) 4.1(9) 5.2(8) 5.5(9) [36]	3.5(9) 4.1(9) 4.2(10) 5.3(8) [36]	4.3(10) 4.5(10) 5.1(8) 5.3(8) [36]	4.3(10) 4.4(10) 5.1(8) 5.2(8) [36]
Pros lab	Morning		3.3(10) 3.5(9) [19]	3.3(10) [10]	3.1(10) 3.5(9) [19]	3.2(10) 3.4(10) [20]
C and B lab	Morning	4.3(10) [10]	4.4(10) [10]			4.5(10) [10]
Study	Morning	4.2(10) [10]	5.3(8) [8]	4.5(10) [10]	4.4(10) [10]	5.4(9) [9]
Dental Surgery	Midday					
Ortho	Midday					
Rest Tutorial	Midday					
OTL	Midday					
Paedo	Midday					
Rest Clinic	Midday					
Pros lab	Midday					
C and B lab	Midday					
Study	Midday					
Dental Surgery	Afternoon	4.4(10) [10]	4.2(10) [10]	4.3(10) [10]	4.5(10) [10]	4.1(9) [9]
Ortho	Afternoon	5.4(9) [9]	5.2(8) [8]	5.3(8) [8]	5.5(9) [9]	5.1(8) [8]
Rest Tutorial	Afternoon	5.5(9) [9]	5.4(9) [9]		5.1(8) [8]	
OTL	Afternoon	3.5(9) 4.1(9) [18]	4.4(10) [10]	3.1(10) [10]	3.3(10) 3.5(9) [19]	3.2(10) 3.3(10) [20]
Paedo	Afternoon	5.3(8) [8]	4.3(10) [10]	4.5(10) [10]	4.4(10) [10]	4.2(10) [10]
Rest Clinic	Afternoon	3.4(5) 4.2(10) 4.3(10) 4.5(10) [35]	3.2(10) 3.3(10) 5.1(8) 5.3(8) [36]	4.4(10) 5.2(8) 5.4(9) 5.5(9) [36]	4.1(9) 4.2(10) 5.2(8) 5.4(9) [36]	4.5(10) 5.3(8) 5.4(9) 5.5(9) [36]
Pros lab	Afternoon		3.1(10) [10]	3.4(10) [10]	3.2(10) [10]	
C and B lab	Afternoon		4.1(9) [9]	4.2(10) [10]		
Study	Afternoon	5.1(8) 5.2(8) [16]	5.5(9) [9]	4.1(9) [9]		4.3(10) [10]

Figure 1: An overview of the 2012 timetable in the deployed system.

Constraints

We discuss the different constraints that are needed to express the requirements expressed in the informal problem description. The first set of constraints links the x and the y variables: as soon as one of the x variables for a time slot t is one, the corresponding y must also be one.

$$\forall_{g\in G}\forall_{s\in S}\forall_{t\in T}\forall_{u\in U}: \quad x_{gstu} \leq y_t$$

The next constraint states that the total number of courses in a subject allocated to a group must be equal to the demand for that group.

$$\forall_{g\in G}\forall_{s\in S}: \quad \sum_{t\in T}\sum_{u\in U} x_{gstu} = \text{demand}_{gs}$$

Each group can only be assigned to one course in any given time period, i.e., the sum of all x variables must be less than or equal to one.

$$\forall_{t\in T}\forall_{g\in G}: \quad \sum_{s\in S}\sum_{u\in U} x_{gstu} \leq 1$$

The important capacity constraint states that for every subject and every time slot, the total number of allocated students, the sum of the group sizes, must be less than or equal to the room capacity plus any extra capacity used. Note that the size of a group differs if it is doubled up.

$$\forall_{s\in S}\forall_{t\in T}: \quad \sum_{g\in G}\sum_{u\in U} x_{gstu}\text{size}_{gu} \leq \text{capacity}_s + v_{st}$$

Groups that cannot be doubled up cannot use the x_{gst1} variables: they must all be zero. As typically only few courses can be doubled up, this dramatically reduces the number of decision variables.

$$\forall_{s\in S}\forall_{g|\neg\text{doubleUp}_g}\forall_{t\in T}: \quad x_{gst1} = 0$$

If on some day d, a group g is assigned to multiple courses that require cleaning, then the z_{dg} variable must be equal to one, incurring the cost for the cleaning in the objective function (1).

$$\forall_{d\in D}\forall_{g\in G}: \quad \sum_{t|\text{onDay}_t=d}\ \sum_{s|\text{needsCleaning}_{gs}}\ \sum_{u\in U} \leq 1 + z_{dg}$$

Allocation Constraints. The following deals with the allocation constraints. If we forbid some time slots for a set of groups on a set of subjects, then we force the corresponding x variables for each member of the sets to zero.

$$\forall_{<\overline{G},\overline{S},\overline{T},\textbf{FORBID}>\in A}\forall_{g\in\overline{G}}\forall_{s\in\overline{S}}\forall_{t\in\overline{T}}\forall_{u\in U}: \quad x_{gstu} = 0$$

If we force the assignment, we cannot directly set some variable to one, as we do not know if we may want to double up for that group. We have to set the sum of two variables to be equal to one, instead.

$$\forall_{<\overline{G},\overline{S},\overline{T},\textbf{FORCE}>\in A}\forall_{g\in\overline{G}}\forall_{s\in\overline{S}}\forall_{t\in\overline{T}}: \quad x_{gst0} + x_{gst1} = 1$$

And finally, if we want to restrict the assignment to a subset of the possible time slots, we can simply force that for any period not in the set, the x variable is set to zero.

$$\forall_{<\overline{G},\overline{S},\overline{T},\textbf{RESTRICT}>\in A}\forall_{g\in\overline{G}}\forall_{s\in\overline{S}}\forall_{t\notin\overline{T}}: \forall_{u\in U}: x_{gstu} = 0$$

If the constraint type is set to DONTCARE, then no constraint is issued.

Incompatibility Subject. For these incompatibility constraints, we state that for any of the subjects s in set $\overline{S}$, either g_1 or g_2 can be assigned in time period t, but not both.

$$\forall_{<\overline{S},G_1,G_2>\in I}\forall_{g_1\in G_1}\forall_{g_2\in G_2}\forall_{s\in\overline{S}}\forall_{t\in T}: \sum_{u\in U} x_{g_1stu} + x_{g_2stu} \leq 1$$

Incompatibility Time Slot. For the more generic time slot incompatibilities, we enforce that for any groups g_1 and g_2, only one of them can be assigned to any subject d at time t.

$$\forall_{<G_1,G_2>\in J}\forall_{g_1\in G_1}\forall_{g_2\in G_2}\forall_{t\in T}: \sum_{s\in S}\sum_{u\in U} x_{g_1stu} + x_{g_2stu} \leq 1$$

Occurrence Constraints. Finally, the occurrence constraints limit the number of courses assigned to a group on a given day to be less than or equal to *limit*.

$$\forall_{<\overline{G},d,\text{limit}>\in O}\forall_{g\in\overline{G}}\forall_{t|\text{onDay}_t=d}: \quad \sum_{s\in S}\sum_{u\in U} x_{gstu} \leq \text{limit}$$

System Evolution and Maintenance

The constraint model underwent significant changes over the three years of operation. The most obvious change was the introduction of the fourth index u for the x_{gstu} decision variables to indicate whether a group is doubled up or not. This facilitated the automated choice of using doubling when required. Before, this constraint could be handled by creating new groups for the doubled up case, with a manual choice which group should be used in which scenario. The new model is more flexible, but the change affected nearly every constraint in the system. The cleaning constraints and the corresponding introduction of the z_{dg} variables were only added in 2012, when this requirement was first expressed by a stakeholder at the dental school.

Also, a number of constraints used in the early model are no longer used. In 2010 a significant number of allocation constraints were specified to FORCE or PRECLUDE the assignment of some courses for specific time periods. Many of these constraints are no longer present, as the more constrained problem now no longer allows these extra preferences. Overlapping groups were used to model the doubling up scenario, this in turn required the Incompatibility Time Slot constraints to avoid overbooking.

Perhaps the most significant change is not in the form of the constraints, but in the tightness of the resource limit. In 2010, there was a demand for $53+3*36+4*39 = 307$ student sessions for the Restorative Clinic, while $10*34 = 340$ sessions were available in AM and PM time slots. In 2012, $49+3*49+4*42 = 364$ sessions were needed, but capacity was limited to $36 * 10 = 360$ slots. This over-constrained problem could only be solved by either doubling up some courses, or adding more seats in the clinic. Either relaxation will only be used when absolutely necessary, so that the Restorative Clinic became a tight resource constraint.

Conclusions

We presented a novel constraint-based timetabling system for dental training schools. The system was developed in collaboration with the dental school at Cork University Hospital, and has been in use since 2010. It has enabled the dental school to meet with a challenging set of demands in terms of student numbers which, without an automated timetabling system, would not have been possible for them to achieve.

In addition to being a novel deployed application, the system is new from a scientific perspective since dental school-like timetabling problems have not been previously studied and reported in the literature. Unlike most education-related timetabling problems which have graph coloring as a challenging core problem, dental school timetabling problems are characterized by challenging bin packing problems.

Acknowledgements

This work was supported by Science Foundation Ireland (Grant Number 05/IN.1/I886s2). The authors would like to thank Professor Robert McConnell and Dr. Francis Burke of the Cork University Hospital and Dental School for the help and support given during the execution of this project.

References

Bonutti, A.; Cesco, F. D.; Gaspero, L. D.; and Schaerf, A. 2012. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals OR* 194(1):59–70.

Burke, E. K.; Pham, N.; Qu, R.; and Yellen, J. 2012. Linear combinations of heuristics for examination timetabling. *Annals OR* 194(1):89–109.

Cambazard, H., and O'Sullivan, B. 2010. Propagating the bin packing constraint using linear programming. In Cohen, D., ed., *CP*, volume 6308 of *Lecture Notes in Computer Science*, 129–136. Springer.

Cambazard, H.; Demazeau, F.; Jussien, N.; and David, P. 2004. Interactively solving school timetabling problems using extensions of constraint programming. In Burke, E. K., and Trick, M. A., eds., *PATAT*, volume 3616 of *Lecture Notes in Computer Science*, 190–207. Springer.

Cambazard, H.; Hebrard, E.; O'Sullivan, B.; and Papadopoulos, A. 2012. Local search and constraint programming for the post enrolment-based course timetabling problem. *Annals OR* 194(1):111–135.

Kingston, J. H. 2012. Resource assignment in high school timetabling. *Annals OR* 194(1):241–254.

McCollum, B.; Schaerf, A.; Paechter, B.; McMullan, P.; Lewis, R.; Parkes, A. J.; Gaspero, L. D.; Qu, R.; and Burke, E. K. 2010. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing* 22(1):120–130.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in csps. In Hayes-Roth, B., and Korf, R. E., eds., *AAAI*, 362–367. AAAI Press / The MIT Press.